Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Technology 4 (2012) 50 – 57

Procedia Technology

Differential Evolution Based Optimization of SVM Parameters for Meta Classifier Design

Tapas Bhadra[a], Sanghamitra Bandyopadhyay[a,*], Ujjwal Maulik[b]

[a]*Machine Intelligence Unit, Indian Statistical Institute, 203, B. T. Road, Kolkata - 700108, India*
[b]*Department of Computer Science and Engineering, Jadavpur University, 188, Raja S C Mallick Road, Kolkata-700032, India*

Abstract

In this paper, we have devised a meta classifier model by simultaneously optimizing different evaluation criteria of classifier performance. For this purpose, a support vector machine (SVM) is used as the underlying classifier and its kernel parameters are optimized using differential evolution. We have also formulated a new fitness function combining different classifier evaluation criteria, i.e., accuracy, sensitivity and specificity. The performance of the proposed meta classification approach is demonstrated to be superior to those of the individual classifiers and also several other meta classifiers based on analyses on three real-life datasets.

Keywords: Differential Evolution, Meta Classifier, Support Vector Machines.

1. Introduction

In pattern recognition, simple classifiers like neural networks, support vector machines (SVMs), K-NNs, decision trees, etc. can predict the class labels of unknown samples by learning from the training data [1]. However, it is seen that all these classifiers do not always give the best classification accuracy for all the problems. Occasionally, to achieve the best accuracy on more complex problems, several base classifiers are run and the final class label is predicted based on different strategies of combination applied to the individual results. This is known as the ensemble of classifiers, or synonymously classifier fusion, combination/aggregation of multiple classifiers, meta classifier, etc. [2]. Some of the common ensemble methods are bagging, boosting, behavioral knowledge space, decision template, and so on. These are distinguished by a function that in principle takes the output of individual classifiers as input and gives a class label as output based on that function. There exists a special kind of ensembling approach called meta classification where a higher level classifier is trained on the outputs of the base level classifiers. In this paper, we have proposed a meta classification scheme based on SVMs, where differential evolution (DE) [3, 4] is used for parameter optimization of the base as well as of the meta SVM classifier. For this optimization purpose, we have devised a new fitness function combining three classifier evaluation criteria, namely, accuracy, sensitivity and specificity. The proposed approach is found to perform better than the individual base classifiers

*Corresponding author
Email addresses: tapas.bhadra@isical.ac.in (Tapas Bhadra), sanghami@isical.ac.in (Sanghamitra Bandyopadhyay), umaulik@cse.jdvu.ac.in (Ujjwal Maulik)

doi:10.1016/j.protcy.2012.05.006

in terms of three evaluation criteria- accuracy, sensitivity and specificity. Performance comparison is also conducted with some other techniques of classifier combination.

2. Related Works

The approaches for ensembling of classifiers can be broadly categorized into three different groups: (i) Decision combination – where same training set is used by different base classifiers at base level, (ii) Divide and conquer – where completely different training sets are used by individual base classifiers, and (iii) Meta classifier – where another level of classifier is needed considering the outputs of the first level base classifiers as features. In the following, we describe these in more details.

2.1. Decision Combination

Decision combination is a very old strategy among the three aforementioned groups. In this category, several classifiers are run on the same training data at the low level. Then, at the next higher level several strategies are taken. Some of them are stated below.

Majority Vote: Let $\{c_1, c_2, \ldots, c_L\}$ be the crisp class labels assigned to a unknown instance x by the classifiers $D_1, D_2, \ldots, D_L$, respectively. Then x is assigned to class label c_j if $count(c_j) > count(c_i)$, $\forall c_i \in \{c_1, \ldots, c_c\} - \{c_j\}$, resolving conflicts by random choice. For this kind of fusion, no parameter tuning is required to be performed on the training data.

Weighted Majority Vote: Majority vote gives equal priority to each of the base classifiers. This is unexpected that all of the individual classifiers give equally good performance on all the datasets. This problem is tackled in weighted majority vote by using weighted parameters for each of the classifiers found by suitable training on the trained dataset.

Behavior Knowledge Space: This kind of classifier estimates the prior probabilities of all class label combinations against the number of classifiers and prepares a look-up table (BKS table) from the training set [5]. Now, given an unknown instance x and its class labels $\{c_1, \ldots, c_L\}$ predicted by L different classifiers, the final class label is predicted based on this BKS table.

Combination Function: Combination function approach uses various functions like min, max, average, product, etc. to calculate the support for every classes $\{w_1, \ldots, w_c\}$ [6]. The class label having maximum support is assigned to the unknown instance x. Let L classifiers $\{D_1, \ldots, D_L\}$ are run on an unknown instance x and we get an $L \times c$ matrix where each classifier $D_i, \forall i = 1, \ldots, L$ may give either crisp or fuzzy or possibilistic class label. Then, the support for class w_j is computed based on the following formula: $\mu_D^j(x) = F(d_{1,j}(x), \ldots, d_{L,i}(x)), \forall j = 1, \ldots, c$ where F() is a l-place operator like min, max, average, product, etc. Finally x is assigned the class label w_j iff $\mu_D^j(x) = \max_k\{\mu_D^k(x)\}$.

2.2. Divide and Conquer

Here, initially the training data is partitioned into different subsets based on random sampling according to either uniform probability or non-uniform probability distribution. There are two popular approaches.

Bagging: Bagging is a bootstrap aggregating technique to improve the classification performance by combining classification results done on several randomly generated training data [7]. Bagging generates m new training sets D_i of size n' from the training data D of size n ($n' < n$) by sampling examples uniformly and with replacement. By sampling with replacement, it is expected that some examples will be appeared multiple times while some will be absent in each D_i. If $n' = n$, then for very large value of n the training set D_i is expected to have 63.2% of the unique examples selected from D. Then m classifiers are run on the above m bootstrap samples to get m different models. For an unknown test example x, the above m models are run and the m results are combined by majority voting to get the unique class label.

Boosting: Boosting is an iterative method to adaptively vary the distribution of the training data by giving more preference on previously misclassified data. At the start of the boosting, all data have the same weights. But, dissimilar to bagging, the weights may differ at the end of each boosting round: data that are correctly classified will get lesser weights while data that are wrongly classified will assign larger weights as compared to weights they have at start of that boosting round. AdaBoost [8] and LogitBoost [9] are two familiar boosting algorithms.

2.3. Meta Classifier

This kind of ensemble of classifiers follow a two-layered architecture where several base classifiers are trained on the training dataset at the first level. Subsequently a meta-classifier is trained on the outputs of the first level classifiers to generate the final prediction in the next level [10]. The meta classifier uses a unique learning algorithm whereas each base classifiers are created by applying different learning algorithms. The meta-classifier tries to learn the relationships between the base classifier predictions and the original class label. Meta classification approach can be implemented in different ways depending on how the meta training data is created. The meta training data can be generated either by appending the individual classifier predictions with original feature set or by taking only the individual classifier results. We have used the latter approach for creating the meta training data.

3. Proposed Meta Classifier Approach

SVM is relatively a new member in the family of classification and regression techniques which originated from statistical learning theory [11]. It has revealed numerous promising results as compared to other well known classifiers in solving many real-life problems. There are two kinds of SVM, namely, non-linear SVM and linear SVM based on whether we need to transform the data into higher dimension or not, respectively. Linear SVM can be further divided into two groups based on whether a linear separable hyperplane can be drawn to classify all training samples without giving any misclassification error or there exists a linear separable hyperplane at the cost of some training errors. The former is called linear SVM for separable case while the latter one is known as linear SVM for non-separable case.

Let, the training data be $\{x_i, y_i\}, i = 1, \ldots, N$, where $x_i \in R^d$ is a vector in d-dimensions feature space and $y_i \in \{-1, +1\}$ be the corresponding class label of x_i.

Linear SVM for separable case looks for the linear separating hyperplane with the largest margin by solving the following quadratic optimization problem:

$$\min_{w,b} \frac{1}{2} w^T w$$

subject to $y_i(w^T x_i + b) \geq 1, i = 1, \ldots, N$.

There are many dataset where a linearly separating hyperplane does not exist to classify all the training samples. To overcome this situation in the form of allowing some errors on the training data, a set of slack variables are introduced. This is known as linear SVM for non-separable case that solves the following quadratic optimization problem:

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i$$

subject to $y_i(w^T x_i + b) \geq 1 - \xi_i, \xi_i \geq 0, i = 1, \ldots, N$, where C is the margin parameter to control the trade-off between the maximization of the margin and minimization of the classification error.

On the other hand, non-linear SVM transforms the input data into a higher dimensional feature space and then finding out a linearly separating hyperplane having the maximal margin. Non-linear SVM solves the following quadratic optimization problem:

$$\min_{w,b} \frac{1}{2} w^T w$$

subject to $y_i(w^T \phi(x_i) + b) \geq 1, i = 1, \ldots, N$, where training data are mapped into a higher dimensional feature space by the function $\phi()$.

We have used here a meta classification scheme that is solely based on SVM having four different kernels (linear, RBF, polynomial and sigmoid) working as the base classifiers.
The different forms of these four kernels are mentioned below:

• **Linear Kernels:** $K(u, v) = u' \star v$.
• **Polynomial Kernel:** $K(u, v) = (gamma \star u' \star v + coeff)^{degree}$.
• **Radial Basis Function Kernel:** $K(u, v) = exp(-gamma \star |u - v|^2)$, for $gamma > 0$.
• **Sigmoid Kernel:** $K(u, v) = tanh(gamma \star |u - v|^2 + coeff)$, for $gamma > 0$,

where K() be the kernel function that takes two feature vectors u and v as input and gives their similarity value in feature space as output. Here, c, the only one SVM parameter, is used to control the trade-off between the training error and the margin, while other parameters, i.e., $gamma$, $degree$ and $coeff$, are all kernel parameters. For all of the above four kernels, the selection of suitable value of c as well as that of the other kernel specific parameters play an important role in giving the optimum classification result.

Selection of parameter values specific to a kernel is a major factor in classification with SVM. We have used a DE-based parameter optimization process to choose best parameters for the classifiers, both at the base level and meta level. DE, proposed by Storn and Price, is a population-based optimization technique in the real search space. The suitable parameter values not only make the classification problem robust but also ensure that the corresponding kernel based SVM will perform equally well on previously unseen examples. For base as well meta level classifiers, the kernel specific parameters are tuned in such a way that three evaluation criteria, i.e., accuracy, sensitivity and specificity, of a classifier performance are simultaneously optimized. (These three evaluation criteria have been defined in Section 4.2.) For this purpose, we have used here a single-objective differential evolution (DE) where these three criteria are combined into the single objective function f() that needs to be maximized. The DE-based optimization technique has been shown in Algorithm 1. This DE-based optimization algorithm is composed of four fundamental phrases, namely, initialization, mutation, crossover and selection. Initialization phrase randomly initializes a population of NP number of D-dimensional individuals of the form :$\{x_{i,1}, \ldots, x_{NP,1}\}$ with $x_{i,1} = \{x^1_{i,1}, \ldots, x^D_{i,1}\}$, $i = 1, \ldots, NP$. Mutation phrase creates a mutated vector $v_{i,g} = \{v^1_{i,g}, \ldots, v^D_{i,g}\}$ and crossover phrase generates a trial vector $u_{i,g} = \{u^1_{i,g}, \ldots, u^D_{i,g}\}$, corresponding to each target vector $x_{i,g}$, $i = 1, \ldots, NP$. Selection phrase compares each trail vector $u_{i,g}$ against the corresponding target vector $x_{i,g}$, $i = 1, \ldots, NP$, of the g^{th} generation and generates the NP target vector $x_{i,g+1}$ for the next generation. This phrase also obtains the parameter vector x_{best} with best fitness value at the end of g^{th} generation. The last three phrases, namely, mutation, crossover and selection, execute sequentially and those repeat as a single block for the maximum number of generations, i.e., g_{max} times. Finally, the best parameter vector x_{best} is obtained after the end of g_{max} generations.

We have utilized this DE-based optimization technique in our proposed meta classification scheme. The detailed methodology of the proposed approach is described in Algorithm 2.

4. Experiments with Meta Classifier

To empirically evaluate the performance of the proposed SVM based meta classifier ($SVMDE_{Meta}$) with other different classifiers such as SVM with linear kernel ($SVMDE_{Lin}$), SVM with RBF kernel ($SVMDE_{RBF}$), SVM with polynomial kernel ($SVMDE_{Poly}$) and SVM with Sigmoid kernel ($SVMDE_{Sig}$), experiments have been carried out on three real-life datasets. For the first dataset, we have used the same training and testing data as explained in [12] while for last two datasets, we have divided each into training and testing data. In the course of these experiments, we have used the linear, RBF, polynomial and sigmoid kernels as implemented within LIBSVM [13].

4.1. Datasets Used

We have collected data from a recently published paper proposing a methodology (TargetMiner) for the prediction of microRNA targets [12]. A part of the data used in this analysis includes several samples of microRNA-mRNA pairs and their sequence features with class labels as “target” and “non-target”. The other two datasets, the *Australian* and *Pima Indians* datasets, are collected from the repository of the University of California at Irvine [14]. The characteristics of these three datasets are summarized in Table 1.

Algorithm 1 DE-based Optimization

function DEOptimization($DataSet$**,** D**,** NP**,** g_{max}**,** F**,** CR**)** where DataSet is the name of the dataset, D is the number of parameters to be optimized, NP is the size of the population, g_{max} is the maximum number of generations, $F \in [0, 1]$ be the scaling factor and $CR \in [0, 1]$ be the crossover rate.

Input: $Dataset$, D, NP, g_{max}, F, CR.

Output: x_{best}: the best parameter vector after g_{max} generations.

Algorithm:

/*** ***Initialization*** ***/
for $i \leftarrow 1$ to NP **do**
  for $j \leftarrow 1$ to D **do**
    $x^j_{i,1} \leftarrow rand(0, 1).(b_{j,U} - b_{j,L}) + b_{j,L}$
    $j \leftarrow j + 1$
  end for
  $i \leftarrow i + 1$
end for
for $g \leftarrow 1$ to g_{max} **do**
  /*** ***Mutation*** ***/
  for $i \leftarrow 1$ to NP **do**
    Randomly generate three integer numbers $r_1, r_2, r_3 \in [1 : NP]$
    where $i \neq r_1 \neq r_2 \neq r_3$
    $v_{i,g} \leftarrow x_{r_1,g} + F.(x_{r_2,g} - x_{r_3,g})$
    $i \leftarrow i + 1$
  end for
  /*** ***Crossover*** ***/
  for $i \leftarrow 1$ to NP **do**
    $jrand \leftarrow floor(D.rand(0, 1))$
    for $j \leftarrow 1$ to D **do**
      if $(rand(0, 1) \leq CR$ **or** $j = jrand)$ **then**
        $u^j_{i,g} \leftarrow v^j_{i,g}$
      else
        $u^j_{i,g} \leftarrow x^j_{i,g}$
      end if
      $j \leftarrow j + 1$
    end for
    $i \leftarrow i + 1$
  end for
  /*** ***Selection*** ***/
  for $i \leftarrow 1$ to NP **do**
    if $f(DataSet, u_{i,g}) \geq f(DataSet, x_{i,g})$ **then**
      $x_{i,g+1} \leftarrow u_{i,g}$
      if $f(DataSet, u_{i,g}) > f(DataSet, x_{best})$ **then**
        $x_{best} \leftarrow u_{i,g}$
      end if
    else
      $x_{i,g+1} \leftarrow x_{i,g}$
    end if
    $i \leftarrow i + 1$
  end for
end for
return(x_{best})

****

function f(DataSet, x)

where $DataSet$ is the name of the dataset and x is the parameter vector.

Input: $Dataset$, x.

Output: $FitnessVal$.

Algorithm:

/*** SVM having parameter vector x is run on the DataSet and subsequently 10-fold accuracy, 10-fold sensitivity and 10-fold specificity are calculated.***/
$FitnessVal = accuracy(DataSet, x) - abs(sensitivity(DataSet, x) - specificity(DataSet, x))$
return($FitnessVal$)

Algorithm 2 DE-based Meta Classification Approach

Step 1: The DE-based optimization scheme provided in Algorithm 1 is applied to get the best parameter values corresponding to each of the four kernel-based SVM classifiers working at the base level.
Step 2: The base classifiers are run on the training data using the kernel-specific parameter values obtained in Step 1 and subsequently four SVM models are generated. Here, training is performed in such a way that it gives a model for the probability estimates of classification.
Step 3: The test data is predicted using the above four SVM models and the accuracy, sensitivity, and specificity values are calculated.
Step 4: The probabilistic scores of the training data, predicted using the aforementioned four kernel-based SVM classifiers, are appended with the original class labels of the same training data to get the corresponding meta training data.
Step 5: The probabilistic scores of the testing data, predicted using the aforementioned four kernel-based SVM classifiers, are appended with the original class labels of the same testing data to get the corresponding meta testing data.
Step 6: For the meta training data, the parameter values corresponding to each of the aforementioned kernel-based SVM classifiers are tuned using the same DE approach. The kernel-based SVM classifier, for which the value of the optimization criterion of the DE approach is maximum, is selected as the best meta classifier.
Step 7: Finally, the meta testing data is predicted using the kernel-specific parameters of the best meta classifier found in Step 6.

Dataset	#Examples	#Features	#Classes
TargetMiner	578(train)/246(test)	30	2
Australian	690	14	2
Pima Indians	768	8	2

Table 1. Characteristics of the datasets used.

4.2. Evaluation Criteria

Three evaluation criteria, i.e., accuracy (Acc), sensitivity (Sn) and specificity (Sp) are considered to show the effectiveness of the proposed SVM-based meta classification scheme and for comparing with the other well-known techniques of ensemble of classifiers. They are defined as follows:
$Acc = \frac{TP+TN}{(TP+FP+TN+FN)}$, $Sn = \frac{TP}{(TP+FN)}$ and $Sp = \frac{TN}{(TN+FP)}$, where TP, TN, FP and FN are the number of correctly predicted positive examples, the number of correctly predicted negative examples, the number of wrongly predicted positive examples and the number of wrongly predicted negative examples, respectively.

4.3. Experimental Results

Two phases of experimentation is carried out for the empirical analysis. In the first phase, the proposed algorithm, $SVMDE_{Meta}$, is compared with the individual base classifiers, i.e., $SVMDE_{Lin}$, $SVMDE_{RBF}$, $SVMDE_{Poly}$ and $SVMDE_{Sig}$. Table 2 lists the accuracy, sensitivity and specificity values of classification obtained from the results on the test data. For the *TargetMiner* data, $SVMDE_{Meta}$ wins over all the individual base classifiers in terms of accuracy and sensitivity, but $SVMDE_{Lin}$ supersedes the others in terms of specificity. In case of the second data *Australian*, we observe that the performance of $SVMDE_{Meta}$ is far better than that of the individual base classifiers in terms of accuracy and specificity, while $SVMDE_{Poly}$ provides better result in terms of sensitivity only. Though for the third data *Pima Indians*, the criterion specificity is better for $SVMDE_{Lin}$ than that for our proposed method, yet the another important criterion, i.e., sensitivity of all the individual base classifiers is much poorer than that of our proposed classification scheme $SVMDE_{Meta}$. Therefore, the proposed meta classification scheme appears to be effective than the individual classifiers in terms of the three criteria, namely, accuracy, sensitivity and specificity.

In the next stage, the performance of the proposed meta classification scheme, $SVMDE_{Meta}$, is compared with some of the existing methods of classifier ensemble such as bagging, AdaBoost, LogicBoost, classification via clustering (CVC) and ensemble selection. The accuracy, sensitivity and specificity of the classification obtained for each test data respectively, have been listed in Table 3. For the *TargetMiner* data, our proposed approach performs the best over all other meta classifiers in terms of accuracy (= 74.39%) and

Dataset	Method	Evaluation Criteria		
		Acc	Sn	Sp
TargetMiner	$SVMDE_{Lin}$	71.54	72.73	**67.80**
	$SVMDE_{RBF}$	70.73	72.19	66.10
	$SVMDE_{Poly}$	72.76	74.87	66.10
	$SVMDE_{Sig}$	71.54	73.26	66.10
	$SVMDE_{Meta}$	**74.39**	**77.01**	66.10
Australian	$SVMDE_{Lin}$	87	90	84
	$SVMDE_{RBF}$	91	92	90
	$SVMDE_{Poly}$	88	**96**	80
	$SVMDE_{Sig}$	81	90	72
	$SVMDE_{Meta}$	**93**	92	**94**
Pima Indians	$SVMDE_{Lin}$	82	55.17	**92.96**
	$SVMDE_{RBF}$	79	58.62	87.32
	$SVMDE_{Poly}$	81	58.62	90.14
	$SVMDE_{Sig}$	79	55.17	88.73
	$SVMDE_{Meta}$	**83**	**75.86**	85.92

Table 2. Comparison of accuracy, sensitivity and specificity of different datasets using $SVMDE_{Lin}$, $SVMDE_{RBF}$, $SVMDE_{Poly}$, $SVMDE_{Sig}$ and $SVMDE_{Meta}$. The best results corresponding to each test dataset is shown in bold.

specificity (= 66.10%) whereas specificity (= 66.10%) obtained using $SVMDE_{Meta}$ remains collectively best with CVC. Herein one needs to note that although the criterion, sensitivity of ensemble selection (= 82.35%) is better than that of $SVMDE_{Meta}$, the other important criterion, specificity of ensemble selection (= 35.59%) is very less. So, for the *TargetMiner* data, our proposed method appears to be the best among all the ensemble classifiers based on the three criteria. In case of the second data *Australian*, $SVMDE_{Meta}$ outperforms the other meta classification techniques, mentioned above, in terms of the three criteria. For the last data *Pima Indians*, $SVMDE_{Meta}$ can not be said to be outperforming the other meta classifiers in terms of single criterion but, based on composite impact of all the three criteria, it wins over others. For example, AdaBoost provides the best Sn value of 96.51%, but its Sp is only 56.34%. Again, the Sn for CVC is 89.66%, but its Sp is only 2.82%. Ensemble selection has the best Sp of 91.55%, but its Sn is only 48.28%. In contrast, the proposed method has good values of Acc (=83%), Sn (=75.86%) and Sp (=85.92%), indicating its effectiveness. As an overall, considering three criteria our proposed method seems to be best among the above mentioned ensemble classifiers.

5. Conclusion

Recently, Support Vector Machine (SVM) has drawn much attention as an efficient technique for solving several real-life classification problems. However, the performance of SVM is sensitive to the selection of the appropriate kernel function and the parameter values associated with that kernel. Thus, to obtain optimal performance for the classification problem, time-consuming grid search is always necessary. In this paper, we have proposed a new kernel function by taking the weighted combination of two existing well-known kernel functions, namely, RBF kernel and polynomial kernel. The performance of the proposed kernel based SVM is established to be better than those of the other well-known kernel based SVM classifiers as well as some of the other state-of-art classifiers based on analyses on forty real-life datasets.

6. Acknowledgement

The authors gratefully acknowledge Department of Science and Technology, India (Grant No. DST/INT/MEX/RPO-04/2008(ii)) for partially supporting this work.

Dataset	Method	Evaluation Criteria		
		Acc	Sn	Sp
TargetMiner	$SVMDE_{Meta}$	**74.39**	77.01	**66.10**
	Bagging	62.60	68.98	42.37
	AdaBoost	67.48	74.33	45.76
	LogicBoost	70.32	80.21	38.98
	CVC	66.67	66.84	**66.10**
	EnsembleSelection	71.14	**82.35**	35.59
Australian	$SVMDE_{Meta}$	**93**	**92**	**94**
	Bagging	87	87.50	86.54
	AdaBoost	87	87.50	86.54
	LogicBoost	86	91.67	80.77
	CVC	79	66.67	90.38
	EnsembleSelection	83	87.50	78.85
Pima Indians	$SVMDE_{Meta}$	83	75.86	85.92
	Bagging	**85**	72.41	90.14
	AdaBoost	68	**96.51**	56.34
	LogicBoost	79	51.72	90.14
	CVC	28	89.66	2.82
	EnsembleSelection	79	48.28	**91.55**

Table 3. Comparison of accuracy, sensitivity and specificity of different datasets using $SVMDE_{Meta}$, Bagging, AdaBoost, LogicBoost, Classification via Clustering (CVC) and Ensemble Selection. The best results corresponding to each test dataset is shown in bold.

References

[1] R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification, 2nd Edition, Springer, New York, 2006.

[2] L. Rokach, Ensemble-based classifiers, Artificial Intelligence Review 33 (2010) 1–39.

[3] R. Storn, K. Price, Differential evolution - a simple and efficient heuristic strategy for global optimization over continuous spaces, Tech. rep., International Computer Science Institute, Berkeley, California.

[4] R. Storn, K. Price, Differential evolution - a simple and efficient heuristic strategy for global optimization over continuous spaces, Journal of Global Optimization 11 (1997) 341–359, [Online]. Available: http://http.icsi.berkeley.edu/~storn/litera.html.

[5] Y. S. Huang, C. Y. Suen, A method of combining multiple experts for the recognition of unconstrained handwritten numerals, IEEE Transactions On Pattern Analysis And Machine Intelligence 17 (1) (1995) 90–93.

[6] L. M. Kuncheva, J. C. Bezdek, R. P. W. Duin, Decision templates for multiple classifier fusion: An experimental comparison, Pattern Recognition 34 (2) (2001) 299–314.

[7] L. Breiman, Bagging predictors, Machine Learning 24 (1996) 123–140.

[8] Y. Freund, R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of Computer and System Sciences 55 (1997) 119–139.

[9] J. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: A statistical view of boosting, The Annals of Statistics 28 (2) (2000) 37–374.

[10] W. H. Lin, R. Jin., A. Hauptmann, Meta-classification of multimedia classifiers, in: International Workshop on Knowledge Discovery in Multimedia and Complex Data, Taipei, Taiwan, 2002.

[11] V. N. Vapnik, The Nature of Statistical Laerning Theory, first edition Edition, Springer, New York, 1995.

[12] S. Bandyopadhyay, R. Mitra, TargetMiner: MicroRNA target prediction with systematic identification of tissue specific negative examples, Bioinfromatics 25 (20) (2009) 2625–2631.

[13] C. C. Chang, C. J. Lin, LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology 2 (3) (2011) 27:1–27:27, Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[14] A. Frank, A. Asuncion, UCI Machine Learning Repository [`http://archive.ics.uci.edu/ml`], University of California, Irvine, School of Information and Computer Sciences (2010).